Cypher-Abfragen Jewish PID

Arbeitsstand: 12. April 2024

Abfragen zu einzelnen Personen

Die Namen und pids sind hier Platzhalter und müssen mit den zu suchenden pids bzw. Namen ausgetauscht werden!

// Eine Person abfragen:

```
Match (n{pid:'903'}) Return n
```

// Nach Nachname mit bestimmten Vorname suchen:

```
Match (n{nachname:'Oppenheimer'}) Where n.name Contains 'Jutta' Return n
```

// Nach Nachname mit Doppelvorname suchen:

```
Match (n{nachname:'Leinen'}) Where n.name = 'David Herz' Return n
```

// Ehepartner von bestimmter Person

```
Match (n1{pid:'5571'})-[VERHEIRATET_MIT]->(n2:Person) Return *
```

Nach mehreren Personen mit bestimmten Merkmalen suchen

// Nach Nachname mit Begräbnisort suchen:

```
Match (n{nachname:'Fränkel'}) Where n.begraebnisort Contains 'Fürth' with n Return n
```

// Nach Nachname mit Wohnort suchen:

```
Match (n{nachname:'Fränkel'})-[r:WOHNORT]->(o:Ort)
WHERE o.name Contains 'Fürth'
Return *
```

// Personen, die in einem Ort begraben sind (Graph):

```
Match (n:Person)-[r:BEGRABEN_IN]->(o:Ort)
```

WHERE o.name = "Fürth"

Return *;

// Personen, die in einem Ort begraben sind (Tabelle):

Match (n:Person)-[r:BEGRABEN_IN]->(o:Ort)

WHERE o.name = "Fürth"

Return n.pid, n.anzeigename, count(*) as Anzahl order by n.pid desc

// Personen, die vor 1740 in einem Ort gestorben sind (Jahr und Ort sind Platzhalter)

MATCH (n:Person) WHERE n.todNorm < "1740"

AND n.begraebnisort contains 'Fürth'

RETURN n.anzeigename, n.todNorm, n.begraebnisort, count(*) as Anzahl order by n.todNorm desc

// Personen, die im Jahr … gestorben sind

MATCH (n:Person) WHERE n.todNorm CONTAINS "1813"

RETURN n.anzeigename, n.todNorm, count(*) as Anzahl order by n.todNorm desc

// Person, unter Obrigkeit von…:

MATCH (n:Person) Where n.obrigkeit='Deutscher Orden' Return n

Kürzester Weg zwischen zwei Personen

// Shortest path

MATCH (n1:Person {pid: '1501' }),(n2:Person {pid: '549' }), p = shortestPath((n1)-[*]-(n2))
Return*

// **Kürzester Weg zwischen zwei Personen mit der Bedingung kein Pfad über Verheiratet mit**

MATCH (n1:Person {pid: '1504' }),(n2:Person {pid: '524' }), p = shortestPath((n1)-[*]-(n2))

WHERE NONE (r IN relationships(p) WHERE type(r)= 'VERHEIRATETMIT')

RETURN p

// **Shortest path mit Ausschluß von zwei Relationships**

```
MATCH (n1:Person {pid: '3699' }),(n2:Person {pid: '3785' }), p = shortestPath((n1)-[*]-(n2))
WHERE NONE (r IN relationships(p) WHERE type(r)= 'BEGRABEN_IN')
AND NONE (r IN relationships(p) WHERE type(r)= 'VERSTORBEN_IN')
RETURN p
```

Gemeindedienste, Tätigkeiten, etc.

// Bestimmter Gemeindedienst in einem bestimmten Ort

```
Match (n:Rolle {name:'Rabbinatsbeisitzer'})-[r:ORT]->(o:Ort)
WHERE o.name Contains 'Fürth'
Return *
```

```
Match (p:Person)-[:GEMEINDEDIENST]->(n:Rolle {name:'Vorsteher'})-[r:ORT]->(o:Ort)
WHERE o.name Contains 'Fürth'
Return p.anzeigename, count(*) as Anzahl order by Anzahl desc
```

Geschlechterspezifische Abfragen

// Wie viele Frauen sind in der Datenbank

```
MATCH (n:Person {g:'f'}) RETURN count(n) as count
```

// Wie viele Frauen mit bestimmten Vornamen sind in der Datenbank

```
Match (p:Person {g:'f'}) Return p.name, count(*) as Anzahl
```

// Wie viele Frauen mit unbekannten Vorname sind in der Datenbank

```
Match (p:Person {g:'f'}) Where p.name = "x" Return p.name, count(*) as Anzahl
```

// Wie viele Männer sind in der Datenbank

```
MATCH (n:Person {g:'m'}) RETURN count(n) as count
```

// Wie viele Männer mit bestimmten Vornamen sind in der Datenbank

Match (p:Person {g:'m'}) Return p.name, count(*) as Anzahl

// Wie viele Männer mit unbekannten Namen sind in der Datenbank

Match (p:Person {g:'f'}) Where p.name = "x" Return p.name, count(*) as Anzahl

// Alle Frauen/Männer, die in Bamberg wohnen (m für Männer unten eintragen)

Match (n:Person {g:'f'})-[w:WOHNORT]->(o:Ort) Where o.name = "Bamberg" Return *

// Migration von Frauen (Alle Frauen, die mindestens 1x ihren Wohnort wechseln**)**

// Frauen als Schutzbriefinhaberinnen (Anzahl)

MATCH (n:Person {g:'f'})-[r:SCHUTZHERR]->(o:Rolle) RETURN count(n) as Anzahl

// Frauen als Schutzbriefinhaberinnen (sortiert nach Orten)

Wirtschaftliche Tätigkeit

// Anzahl wirtschaftlich tätiger Frauen

Match (n:Person {g:'f'})-[t:TAETIGKEIT]->(r:Rolle) return count(n) as count

// Wirtschaftliche Tätigkeit von Frauen (Name und Tätigkeit)

Match (n:Person {g:'f'})-[t:TAETIGKEIT]->(r:Rolle) return n.name, n.nachname, r.name, count(*) as Anzahl order by n.nachname DESC

// Anzahl wirtschaftlich tätiger Männer

Match (n:Person {g:'m'})-[t:TAETIGKEIT]->(r:Rolle) return count(n) as count

// Wirtschaftliche Tätigkeit von Männern (Name und Tätigkeit)

Match (n:Person {g:'m'})-[t:TAETIGKEIT]->(r:Rolle) return n.name, n.nachname, r.name, count(*) as Anzahl order by n.nachname DESC

Eheschließung, Migration und soziale Endogamie

// Verwandtschaftliche Ehen

```
match (p:Person)-[:FAMILIENVERBAND]->(f:Familienverband)
return p.pid, p.anzeigename, count(*) as Anzahl, collect(distinct f.label) order by Anzahl
desc;
```

```
MATCH (e2:Person)-[:KIND_VON]->
(k1:Person)-[:KIND_VON]->(g:Person)
<-[:KIND_VON]-(k2:Person)<-[:KIND_VON]-
(e1:Person)-[:VERHEIRATET_MIT]-(e2)
RETURN * LIMIT 10;
```

// Verbotene Ehen

```
MATCH (e2:Person)-[:KIND_VON]->
(k1:Person)-[:KIND_VON]->(g:Person)
<-[:KIND_VON]-(k2:Person)<-[:KIND_VON]-
(e1:Person {g:'m'})-[:VERHEIRATET_MIT]-(k1 {g:'f'})
RETURN *;
```

// Erlaubte Onkel-Ehen

```
MATCH (e2:Person)-[:KIND_VON]->
(k1:Person)-[:KIND_VON]->(g:Person)
<-[:KIND_VON]-(k2:Person)<-[:KIND_VON]-
(e1:Person {g:'f'})-[:VERHEIRATET_MIT]-(k1 {g:'m'})
RETURN *;
```

Soziale Endogamie

// Gemeindedienst Schwiegervater – Schwiegersohn (genauso mit Taetigkeit)

```
match (kp1:Person)-[:VERHEIRATET_MIT]-(k1:Person {g:'m'})<-[:KIND_VON]-(t:Person)-
[:VERHEIRATET_MIT]-(so:Person)//<-[:KIND_VON]->(k2:Person)-[:VERHEIRATET_MIT]-(kp2)
match (k1)-[:GEMEINDEDIENST]->(g1)-->(r1:Gemeindedienst)
```

match (so)-[:GEMEINDEDIENST]->(g2)-->(r1)
match (o1:Ort)<-[:ORT]-(g1)
match (o2:Ort)<-[:ORT]-(g2)
return distinct k1.anzeigename AS Schwiegervater, g1.name as Rolle1, o1.name as RollenortSchwiegervater, so.anzeigename as Schwiegersohn, r1.name as Rolle2, o2.name as RollenortSchwiegersohn;

// Tätigkeiten Schwiegervater und Schwiegersohn

match (kp1:Person)-[:VERHEIRATET_MIT]-(k1:Person {g:'m'})<-[:KIND_VON]-(t:Person)-[:VERHEIRATET_MIT]-(so:Person)//<-[:KIND_VON]->(k2:Person)-[:VERHEIRATET_MIT]-(kp2)
match (k1)-[:TAETIGKEIT]->(g1)-->(r1:Taetigkeit)
match (so)-[:TAETIGKEIT]->(g2)-->(r1)
match (o1:Ort)<-[:ORT]-(g1)
match (o2:Ort)<-[:ORT]-(g2)
return distinct k1.anzeigename AS Schwiegervater, g1.name as Rolle1, o1.name as RollenortSchwiegervater, so.anzeigename as Schwiegersohn, r1.name as Rolle2, o2.name as RollenortSchwiegersohn;

//Fürth Gemeindedienst Schwiegervater und Schwiegersohn

match (kp1:Person)-[:VERHEIRATET_MIT]-(k1:Person {g:'m'})<-[:KIND_VON]-(t:Person)-[:VERHEIRATET_MIT]-(so:Person)//<-[:KIND_VON]->(k2:Person)-[:VERHEIRATET_MIT]-(kp2)
match (k1)-[:GEMEINDEDIENST]->(g1)-->(r1:Gemeindedienst)
match (so)-[:GEMEINDEDIENST]->(g2)-->(r1)
match (o1:Ort)<-[:ORT]-(g1)
match (o2:Ort)<-[:ORT]-(g2)
where o2.name Contains 'Fürth'
return distinct k1.anzeigename AS Schwiegervater, g1.name as Rolle1, o1.name as RollenortSchwiegervater, so.anzeigename as Schwiegersohn, r1.name as Rolle2, o2.name as RollenortSchwiegersohn;

// Fürth Tätigkeit Schwiegervater und Schwiegersohn

match (kp1:Person)-[:VERHEIRATET_MIT]-(k1:Person {g:'m'})<-[:KIND_VON]-(t:Person)-[:VERHEIRATET_MIT]-(so:Person)//<-[:KIND_VON]->(k2:Person)-[:VERHEIRATET_MIT]-(kp2)

```
match (k1)-[:TAETIGKEIT]->(g1)-->(r1:Taetigkeit)
match (so)-[:TAETIGKEIT]->(g2)-->(r1)
match (o1:Ort)<-[:ORT]-(g1)
match (o2:Ort)<-[:ORT]-(g2)
where o1.name Contains'Fürth'
or o2.name Contains'Fürth'
return distinct k1.anzeigename AS Schwiegervater, g1.name as Rolle1, o1.name as RollenortSchwiegervater, so.anzeigename as Schwiegersohn, r1.name as Rolle2, o2.name as RollenortSchwiegersohn;
```

// Tätigkeiten Väter und Schwiegerväter mind. einer in Fürth

```
match (kp1:Person)-[:VERHEIRATET_MIT]-(k1:Person {g:'m'})<-[:KIND_VON]-(t:Person)-[:VERHEIRATET_MIT]-(so:Person)-[:KIND_VON]->(k2:Person)-[:VERHEIRATET_MIT]-(kp2)
match (k1)-[:TAETIGKEIT]->(g1)-->(r1:Taetigkeit)
match (so)-[:TAETIGKEIT]->(g2)-->(r1)
match (o1:Ort)<-[:ORT]-(g1)
match (o2:Ort)<-[:ORT]-(g2)
where o1.name Contains'Fürth'
or o2.name Contains'Fürth'
return distinct k1.anzeigename AS Schwiegervater, g1.name as Rolle1, o1.name as RollenortSchwiegervater, k2.anzeigename as Vater, r1.name as Rolle2, o2.name as RollenortVater
```

// Tätigkeiten Väter und Schwiegerväter gleich ohne Schwiegermutter (mind. einer in Fürth)

```
match (kp1:Person)-[:VERHEIRATET_MIT]-(k1:Person {g:'m'})<-[:KIND_VON]-(t:Person)-[:VERHEIRATET_MIT]-(so:Person)-[:KIND_VON]->(k2:Person)//-[:VERHEIRATET_MIT]-(kp2)
match (k1)-[:TAETIGKEIT]->(g1)-->(r1:Taetigkeit)
match (so)-[:TAETIGKEIT]->(g2)-->(r1)
match (o1:Ort)<-[:ORT]-(g1)
```

```
match (o2:Ort)<-[:ORT]-(g2)
where o1.name Contains'Fürth'
or o2.name Contains'Fürth'
return distinct k1.anzeigename AS Schwiegervater, g1.name as Rolle1, o1.name as
RollenortSchwiegervater, k2.anzeigename as Vater, r1.name as Rolle2, o2.name as
RollenortVater
```

// Gemeindedienste Väter und Schwiegerväter gleich (mind. einer in Fürth)

```
match (kp1:Person)-[:VERHEIRATET_MIT]-(k1:Person
{g:'m'})<-[:KIND_VON]-(t:Person)-[:VERHEIRATET_MIT]-(so:Person)-[:KIND_VON]-
>(k2:Person)-[:VERHEIRATET_MIT]-(kp2)
match (k1)-[:GEMEINDEDIENST]->(g1)-->(r1:Gemeindedienst)
match (so)-[:GEMEINDEDIENST]->(g2)-->(r1)
match (o1:Ort)<-[:ORT]-(g1)
match (o2:Ort)<-[:ORT]-(g2)
where o1.name Contains'Fürth'
or o2.name Contains'Fürth'
return distinct k1.anzeigename AS Schwiegervater, g1.name as Rolle1, o1.name as
RollenortSchwiegervater, k2.anzeigename as Vater, r1.name as Rolle2, o2.name as
RollenortVater
```

// Gemeindedienste Väter und Schwiegerväter gleich ohne Schwiegermutter (mind. einer in Fürth)

```
match (kp1:Person)-[:VERHEIRATET_MIT]-(k1:Person {g:'m'})<-[:KIND_VON]-(t:Person)-
[:VERHEIRATET_MIT]-(so:Person)-[:KIND_VON]->(k2:Person)//-[:VERHEIRATET_MIT]-(kp2)
match (k1)-[:GEMEINDEDIENST]->(g1)-->(r1:Gemeindedienst)
match (so)-[:GEMEINDEDIENST]->(g2)-->(r1)
match (o1:Ort)<-[:ORT]-(g1)
match (o2:Ort)<-[:ORT]-(g2)
where o1.name Contains'Fürth'
or o2.name Contains'Fürth'
```

return distinct k1.anzeigename AS Schwiegervater, g1.name as Rolle1, o1.name as RollenortSchwiegervater, k2.anzeigename as Vater, r1.name as Rolle2, o2.name as RollenortVater

// Gleicher Gemeindedienst Vater-Sohn mind. einer in Fürth

```
match (p1:Person)-[:KIND_VON]->(p2)
match (p1)-[:GEMEINDEDIENST]->(g1)-->(r1:Gemeindedienst)
match (p2)-[:GEMEINDEDIENST]->(g2)-->(r1)
match (o1:Ort)<-[:ORT]-(g1)
match (o2:Ort)<-[:ORT]-(g2)
where o1.name Contains 'Fürth'
or o2.name Contains 'Fürth'
return distinct p1.anzeigename AS Sohn, g1.name as Rolle1, o1.name as RollenortSohn, p2.anzeigename as Vater, r1.name as Rolle2, o2.name as RollenortVater;
```

// Gleicher Beruf Vater-Sohn mind. einer in Fürth

```
match (p1:Person)-[:KIND_VON]->(p2)
match (p1)-[:TAETIGKEIT]->(g1)-->(r1:Taetigkeit)
match (p2)-[:TAETIGKEIT]->(g2)-->(r1)
match (o1:Ort)<-[:ORT]-(g1)
match (o2:Ort)<-[:ORT]-(g2)
where o1.name Contains 'Fürth'
or o2.name Contains 'Fürth'
return distinct p1.anzeigename AS Sohn, g1.name as Rolle1, o1.name as RollenortSohn, p2.anzeigename as Vater, r1.name as Rolle2, o2.name as RollenortVater;
```

Zur Fehlersuche:

// Kind von sich selbst

```
MATCH (n1:Person)-[r:KIND_VON]-(n1:Person)
RETURN *;
```

// Verheiratet mit sich selbst

```
MATCH (n1:Person)-[r:VERHEIRATET_MIT]-(n1:Person)
RETURN *;
```

// **Zu oft verheiratet**

```
MATCH (p1:Person)-[r:VERHEIRATET_MIT]-(p2:Person)
RETURN p1.pid, p1.name, count(r) as Anzahl ORDER BY Anzahl DESC;
```

// 'Gleichgeschlechtliche Ehen' - (zeigt die falsche Geschlechterbezeichnung an)

```
MATCH (n1:Person)-[r:VERHEIRATET_MIT]-(n2:Person)
WHERE n1.g = n2.g
RETURN *;
```

// **Selbstbezüge**

```
MATCH (n)--(n)
RETURN n.pid, n.anzeigename;
```

// Person1 verheiratet mit Person2 und gleichzeitig Kind von Person2

```
MATCH (n1:Person)-[r1:VERHEIRATET_MIT]-(n2:Person), (n1:Person)-[r2:KIND_VON]-
(n2:Person) RETURN *
```

// Zur Fehlersuche/Normierung von Namen

```
Match (n:Person)
WHERE n.name starts with 'J'
Return Distinct Collect(n.pid), n.name AS vorname ORDER BY vorname Desc
```

// Personen Vorname startet mit "Jen" und Nachname mit "J" sortiert nach Nachname

```
Match (n:Person)
WHERE n.name starts with 'Jen'
AND n.nachname starts with 'J'
Return n.nachname, n.name
```